REMARKS

This amendment is being filed as a response to the Office Action of September 30, 2008. Reconsideration is respectfully requested in view of these clarifying amendments and remarks.

Specification

The Specification has been amended to reflect the status change of Related Applications.

Claim Objections

The Office has objected to claims 2, 10, and 18. Applicants respectfully assert that such objections are rendered moot in view of the amendments to claim 2 submitted hereinabove.

Double Patenting

Claims 1, 2, and 25 were rejected under a nonstatutory double patenting rejection. This rejection is rendered moot in view of the terminal disclaimer filed to obviate the rejection over US Patent No. 7,406,687.

Rejections under 35 USC 101

Claims 19-24 and 26 were rejected under 101 as the claims were directed to a non-statutory subject matter. Applicants respectfully request that such rejection be removed in view of the amendments made hereinabove to claims 19 and 26.

Rejections under 35 USC § 102

Claims 1-14 and 18-26 have been rejected under 35 USC 102(b) as being anticipated by Czajkowski et al. (cited art, "Code sharing among Virtual Machines"). This rejection is respectfully traversed.

Claim 1 defines dividing a runtime representation of the first class type into a first loader independent part and a first loader dependent part (emphasis added). The Office has asserted that this feature is anticipated by Czajkowski in page 12, parg. 2 & 4), excerpted below:

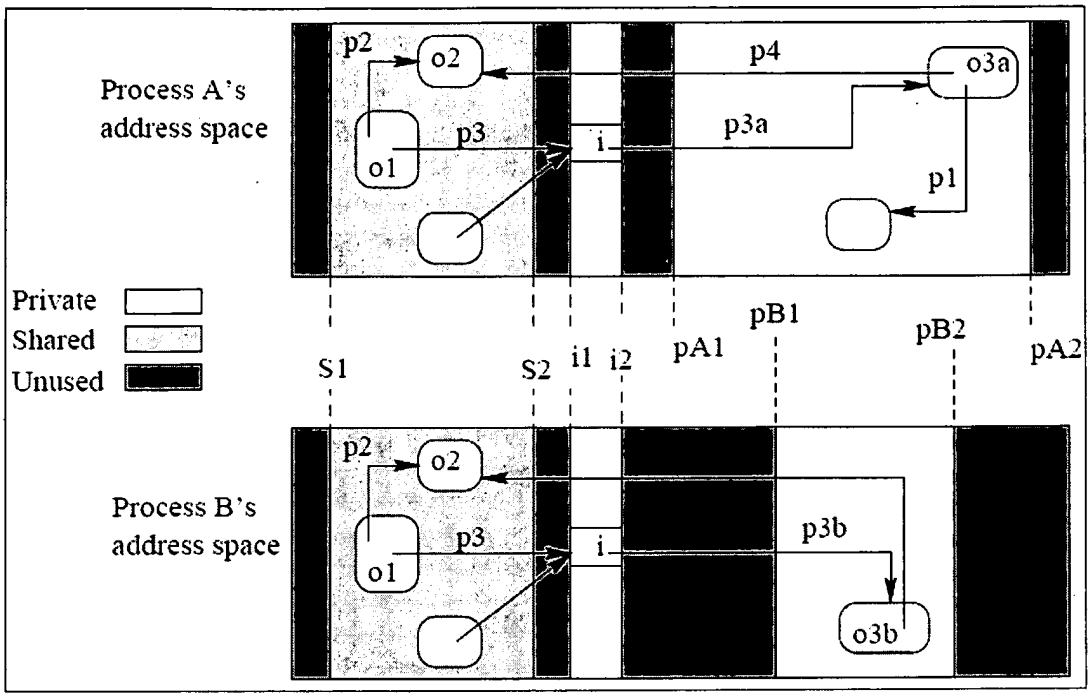
> "For the purpose of comparison with the other approaches to code sharing explored in this paper, only the changes to the runtime representation of classes, to the interpreter, and to the dynamic compiler are described in what follows," and
>
> "Most of the runtime representation of a class is already independent of any particular execution context and can therefore be shared as is. This sharable portion includes the constant pool, debugging information, the descriptions of methods and fields, including information resolved at runtime such as the offset of an instance variable from the beginning of an object or the index of a method in a virtual table, and, given appropriate changes to the interpreter, the bytecode of methods. The runtime constant pool cache (a subset of the runtime constant pool optimized for use by both the interpreter and code produced by the runtime compiler) can also be shared after a few minor modifications" (page 12, parg. 2 & 4 -emphasis added).

Applicants respectfully disagree. There is no teaching in Czajkowski of loader dependent and loader independent runtime representations. The Examiner is reminded that a claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described in a single prior art reference. *Verdegaal Bros. v. Union Oil Co. Of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). Moreover, the identical invention must be shown in as complete detail as contained in the claim. *Richardson v. Suzuki Motor Co.*868 F.2d 1226, 1236, 9USPQ2d 1913, 1920 (Fed. Cir. 1989). The elements must be arranged as required by the claim.

Czajkowski merely teaches runtime representation of a class is "independent of any particular execution context." Further, Czajkowski teaches runtime constant pool optimized for use by both the interpreter and ... the runtime compiler. However, Czajkowski is silent with respect to loader dependent and loader independent runtime representations, thus the Office's rejection is improper.

In addition, claim 1 defines determining whether a runtime representation of the second class type can use the first loader independent part of the runtime representation of the first class type. The Office has asserted that Czajkowski teaches this feature in page 3, third paragraph (excerpted below with related Figure 1.)

> Figure 1 shows how the address space of each process executing ShMVM is divided up into a private area, a shared area, and an indirection area. The last two must be at the same virtual address in all processes. The shared area holds objects that are shared across all processes. The private area contains all data private to a process, including the garbage-collected heap and thread stacks. Objects in any area can reference objects in the shared area directly, using their virtual memory addresses (e.g., pointers p_2 and p_4 holds the address of shared object o_2 in Fig. 1). Objects in the private area of a process can also directly reference objects in the private area of the same process (e.g., p_1). However, pointers to objects of the private area of any process (e.g., pointer p_{3a} in process A, or p_{3b} in process B) cannot be stored in the shared area, since they hold a virtual memory address that may not correspond to the same object in different processes. To solve this problem, each process maintains a private indirection table mapped at the same virtual address (i_1 in Fig. 1): objects in the shared area reference objects in the private area via an entry in the indirection table (an indirection). Addresses to indirections (e.g., p_3 in Fig. 1) are valid across all processes, and therefore, can be stored in shared objects (e.g., o_1). Each indirections holds the virtual address of the object associated with it, which can be different for each process. For instance, shared object o_1 refers to indirection i, which has the same address p_3 in all processes; i holds the address of o_{3a} in process A and of o_{3b} in B.



Applicants respectfully disagree. Czajkowski merely teaches "the address space of each process executing ShMVM is divided up into a private area, a shared area, and an indirection area" (emphasis added). However a reference to process sharing a memory space does not anticipate determining whether a runtime representation of the second class type can use the first loader independent part of the runtime representation of the first class type. There is not even a mention in the excerpt relied upon by the Office to a runtime representation of a class type, much less a reference to determining if a runtime representation of a class type can use another class loader independent part. Thus, the Office's rejection is improper.

Claim 3 defines determining a satisfaction of a first condition, the first condition being defined by a first class file being the same as a second class file used by the second

class loader to define the second class type, which according to the Office is taught by Czajkowski in page 14, 2nd paragraph. Applicants respectfully disagree. Czajkowski teaches that "class initialization barriers are omitted if their target is one of (i) the class defining the method being compiled, (ii) a super-class of the above, (iii) a class initialized at virtual machine startup whose initialization was not triggered by the method being compiled, or (iv) a class for which a barrier has been already emitted upward the instruction stream of the method being compiled" (page 14, 2nd parg.). However, none of these conditions refer to a first class file being the same as a second class file. Thus, Czajkowski does not anticipate determining a satisfaction of a first condition being defined by a first class file being the same as a second class file used by the second class loader to define the second class type, as claimed by Applicants.

Still yet, claim 5 further defines the aforementioned feature of claim 3 and teaches identifying that each byte of the second class file is the same as a corresponding byte of the first class file. The Office has asserted that this feature is taught by Czajkowski in page 3, third paragraph excerpted above. Applicants respectfully disagree.

Czajkowski teaches objects shared by different process, but Czajkowski is silent with respect to files having the same corresponding bytes. Thus, Czajkowski does not teach identifying that each byte of the second class file is the same as a corresponding byte of the first class file, as claimed by Applicants.

In addition, claim 7 defines the feature wherein the reference to the loader independent part of the runtime representation of the super class type of the first class type is stored in the loader dependent part of the runtime representation of the super class type of the first class type, and the reference to the loader independent part of the runtime representation

of the second class type is stored in the loader dependent part of the runtime representation of the super class type of the second class type. The Office has asserted that Czajkowski teaches the features of claim 7 in page 4, 2nd paragraph, excerpted below.

> "Indirections are allocated as needed, when shared data that need to reference private ones are stored in the shared area. A field of a shared object that references a private object either holds a null value or the address to an indirection. JVMs initialize the entries of their indirection table to null. When a JVM uses a shared object for the first time, it initializes the indirections referenced from that object. Indirections may also be initialized lazily, in which case the null value is used to detect whether an indirection has been initialized. The garbage collector running in one process can relocate a private object referenced from the shared area independently of other processes by updating the indirections of its copy of the indirection table. This solution leverages virtual memory to efficiently support the one-to-many mapping between shared and private data" (page 4, 2nd paragraph - emphasis added)."

There is no teaching in the excerpt cited by the examiner of any feature of claim 7, thus the Office's rejection is improper.

Further yet, independent claim 19 defines determining a satisfaction of a first condition, the first condition being defined by a first binary representation of the first software component being the same as a second binary representation of the second software component used by the second component loader to define a second software component type, which is believed to be patentable for at least the same reasons described with respect to claim 3.

In addition, claim 19 defines determining the satisfaction of a third condition, the third condition being defined by the second software component type having the same unimplemented methods as the first software component type. The Office has asserted that this feature is anticipated by Czajkowski in page 14, parg. 2, and page 10, parg 3. Applicants respectfully disagree. There are no teachings of unimplemented methods in the paragraphs

cited by the Examiner, and much less of the second software component type having the same unimplemented methods as the first software component type. Thus, the Office's rejection is improper.

Independent claims 25 and 16 are believed to be patentable for at least the same reasons identified for claims 1 and 19, respectively.

For at least these reasons, claims 1, 3, 5, 7, 19, 25, and 26 are submitted to be patentable over the art of record. Therefore, the Office is respectfully requested to withdraw the §102 rejection, as not all elements are taught by Czajkowski.

Rejections under 35 USC § 103(a)

Claims 15-17 have been rejected under 35 U.S.C. 103(a) as being unpatentable over Czajkowski et al., in view of Skibbie et al. (U.S. Patent No. 6,910,128). This rejection is respectfully traversed. Applicants respectfully request that the rejections of dependent claims 15-17 be withdrawn, as the added references do not cure the deficiencies in the rejections of the independent claims.

The dependent claims are submitted to be patentable for at least the same reasons the independent claims are believed to be patentable. The Applicants therefore respectfully request reconsideration and allowance of the pending claims. A Notice of Allowance is respectfully requested.

If the Examiner has any questions concerning the present amendment, the Examiner is kindly requested to contact the undersigned at (408) 774-6920. If any other fees are due in

connection with filing this amendment, the Commissioner is also authorized to charge Deposit Account No. 50-0805 (Order No. SUNMP337A).

Respectfully submitted,
MARTINE PENILLA & GENCARELLA, LLP

/Jose M. Nunez/

Jose M. Nunez
Reg. No. 59,979

710 Lakeway Drive, Suite 200
Sunnyvale, CA 94085
Telephone: (408) 774-6920
Facsimile: (408) 749-6901
email: jose@mpiplaw.com
Customer Number 32291